Action sequencing in VR, a no-code approach*

Flavien Lécuyer[1], Valérie Gouranton[1], Adrien Reuzeau[1], Ronan Gaugne[2], and Bruno Arnaldi[1]

[1] Univ Rennes, INSA Rennes, Inria, CNRS, IRISA, Rennes, France
[2] Univ Rennes, Inria, CNRS, IRISA, Rennes, France

Abstract. In many domains, it is common to have procedures, with a given sequence of actions to follow. To perform such procedures, virtual reality is a helpful tool as it allows to safely place a user in a given situation as many times as needed, without risk. Indeed, learning in a real situation implies risks for both the studied object – or the patient – (e.g. badly treated injury) and the trainee (e.g. lack of danger awareness). To do this, it is necessary to integrate the procedure in the virtual environment, under the form of a scenario. Creating such a scenario is a difficult task for a domain expert, as the coding skill level needed for that is too high. Often, a developer is needed to manage the creation of the virtual content, with the drawbacks that are implied (e.g. time loss and misunderstandings).
We propose a complete workflow to let the domain expert create their own scenarized content for virtual reality, without any need for coding. This workflow is divided in two steps: first, a new approach is provided to generate a scenario without any code, through the principle of *creating by doing*. Then, efficient methods are provided to reuse the scenario in an application in different ways, for either a human user guided by the scenario, or a virtual actor controlled by it.

1 Introduction

In our daily lives, many procedures imply to follow a specific sequence of actions, in order to reach a good result. Some of those procedures force the user to respect it to the letter, such as for a surgical operation where even the slightest mistake generates critical problems. On the contrary, some of them serve more as indications to guide a user, like a signposted route in a museum. When those situations are recreated in virtual reality, this action sequencing must be made explicit, through the use of a scenario.

Usually, the scenario is designed by a domain expert (i.e. the expert, working in the application domain, such as a teacher, a medical doctor, an archaeologist, etc.). However, due to the need for coding skills to integrate the scenario in virtual reality, the help of a developer is almost always needed. Since the developer needs to understand correctly the needs of the domain expert before being able

* This work is part of the ANR-16-FRQC-0004 INTROSPECT project, and the SUNSET project funded by the ANR-10-LABX-07-01 "Investing for the Future" program

to translate the scenario into machine code, this step is usually time consuming, and error prone. The strong presence of implicit constraints in most domains is also an important source of difficulties in that process. Even when the scenario is obtained, the domain expert needs a way to reuse it. According to the needs of the application (*learn by doing*, *monitor by seeing*, *collaborate by doing* or *learn by seeing*), the integration of the scenario is done quite differently.

A common way of guiding someone through a procedure is to demonstrate it. This way, every action composing the sequence is made explicit, since it i,s directly shown to the observer. Based on this metaphor, we propose a new workflow to create scenarios translating the procedures through a demonstration in a virtual environment. A preliminary version of this work has been reported in [19]. In this paper, we add an efficient way of reusing the generated scenario in the virtual reality application, adapted to different needs. After the analysis of related works in Section 2, we present how a scenario can be easily generated and reused by a domain expert, through the use of efficient metaphors and tools, with five points: the *create by doing* approach for the creation of scenarios, in Section 3; the presentation of a scenario authoring tool based on this approach, in Section 4; the reuse of the scenario with both real users and virtual actors, in Section 5; the description of two use cases demonstrating the creation and reuse of the scenarios, in Section 6; and a short user study in Section 7. This is an extended version of the paper presented at CGI 2019 [19].

2 Related work

For the management of scenarios in virtual reality application, some works proposed a way to model scenarios. Out of those scenario models, the majority is provided without any graphical representation. This limitation, forcing the scenario author to write code, makes it difficult to create new scenarios for a non-computer scientist. On the contrary, graphical representations provide a better abstraction. However, it is to note that this abstraction must be done with caution, so as to not lose expressiveness [13]. Some works propose a graphical representation for the scenarios. Thanks to this, the scenario becomes understandable for the domain experts, who are generally unfamiliar with computer science, but are the ones who know best what is expected of the scenario. The

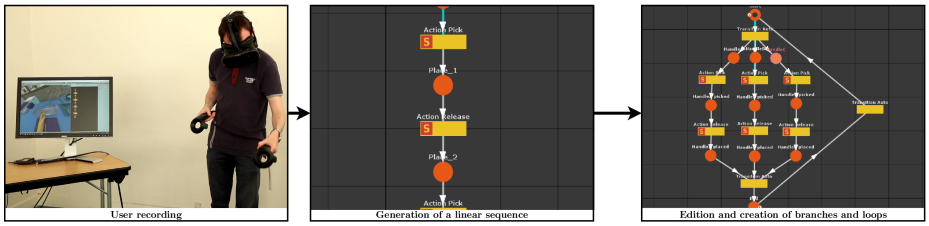

Fig. 1. The proposed workflow starts from the action recording to generate a straight scenario, which can then be edited to obtain a more complex sequence

graphical representation associated with those scenario models highly depends on the kind of representation formalism it is based on. For instance, HAVE [8] uses UML representation, as it is based on activity diagrams. The use of such a representation eases the learning for a developer, and can be understood by an expert with some learning, although it is difficult for someone other than the developer to create the content. Similarly, a Grafcet-like representation is used for LORA++, since it uses similar concepts [12]. This has the main advantage of presenting an easily understandable representation for a domain expert but this requires more effort for the fine tuning of the actions, since the graphical representation is not completely adapted for this. In order to integrate this directly in the model, some made a lower level model, which is close to the scripts used by the application. This is the case for HPTS++ [18], in which the designer needs to write scripts based on finite state machines, as described in the presentation of HCSM [10]. On the other hand, Story Nets proposes to make finite state machines which are very close to the code [5]. The major problem that is encountered is that a regular representation is often not enough to fully represent the concepts we need. This can be seen with IVE [5], in which the concept of place from the Petri nets is derived into two separate concepts (actor and preconditions) to make the representation expressive enough to define a scenario. Instead, it is possible to specialize and extend an existing concept, as ABL does for finite state machines [23], as well as #SEVEN for Petri nets [9]. While #SEVEN focuses more on the interactions in the environment, ABL is designed to define the behaviour of virtual agents. Even though the domain experts can understand such a scenario, the authoring can still be difficult to apprehend. For instance, the logic of how to build a working scenario may still be difficult to apprehend, as it requires a link to the code to be functioning. To help with the authoring, the concept of "*creating by doing*" aims to create the scenarios through the recording of the actions. As an example, the work presented by Angros et al. [2] starts from a recording of the user to generate a first version of a scenario, which is then generalized. To do this, the application creator informs the scenario manager what are the objectives of the scenario to be used. This method is interesting to create short scenarios, with atomic sequences, but the recording of a long scenario is done step by step, require more time.

Another efficient way to define a scenario is to define hierarchical goals. An example of this approach is given in [22], in which the expert can define the main elements of the scenario, before giving more details to integrate it with the virtual environment. This kind of top-down approach is efficient for the creation of such scenarios since it allows the expert to intervene at a higher level of abstraction. However, it is less suited for precise procedures, since the expert may forget to mention some of the actions to perform. Indeed, it is not unusual for the expert to explain the procedure at a high level, with too few details for the implementation. Although it can be fixed by iterating between the expert and the developer, the time required for this could be saved.

In parallel, Some works proposed to provide a replay function in virtual environments, which can be seen as a way to record the user actions to generate

a scenario, or at least a trace of what the user has been doing. For instance, MoCap can be seen as a way to record the user's actions, such as in Chan et al.'s work [7], or Bailenson et al.'s work [3]. Another interesting work is the one done by the MIT Education Arcade with *Mystery at the Museum* (*M@M*) [16] and *Environmental Detectives* [17], in which replaying the previous actions is possible, and provided as a means to remind the players of what already happened. Unfortunately, their approach is not really detailed, and the replay function seems to rely more on a log of the events than on a reusable scenario. A scenario that is generated through the execution can also be reused to generate a final product, as done in [25], where the user can influence the evolution of the story. Those choices are recorded in order to produce a movie taking into account every choice made by the user. This means that the scenario can be reused only in a passive way. A better possibility would be to create a mixed reality application through an immersive environment, as proposed in [20]. However, the creation of a scenario for a virtual reality application is a tedious task for a domain expert.

To conclude to this section, we can observe that the question of simplifying the creation of scenarios for virtual reality has been asked multiple times. To ease the creation of scenarios, several models have been proposed, with visual representations of the scenarios to make them more easily understandable. However either coding skills or a learning phase is needed to be able to use them. On the contrary, hierarchical scenarios are easily understandable, but make the definition of the precise actions more difficult. Other works focused on the principle of creating scenarios through the actions of a user, to reuse the scenario afterwards, with reuses varying from a logging of the past actions to a movie generated from the scenario. Still, none of them reuses the scenario directly in the virtual environment.

3 Scenario creation

The authoring of scenarios for virtual reality applications is a tedious task for both the domain expert and the developer, since both of them hold a part of the necessary knowledge to create it, and the communication between them can be difficult. In this section, we first define some important notions for our approach in Section 3.1, to detail the creation of the scenario in Section 3.2

3.1 Definitions

Before we present in more details our approach, it is important to define what we call a "scenario" in this paper. In our context, a scenario is a sequence of actions linked in a specific way. The links between the actions define a temporal order, to prevent an action from being executed if other ones are not done before. This kind of scenario is particularly useful if the internal states of the objects are not enough to ensure the coherency of the environment. Of course, the scenario can be enriched with loops, to define that a part of the sequence must be repeated a given number of times. It can also contain branches, to let the user choose

between several ways of continuing the scenario, or represent two sequences that may be done in parallel.

To make the *create by doing* possible, we focus on atomic actions, so that they can be considered executed instantly, instead of considering continuous actions such as walk. Indeed, to define the order of actions, the fact that an action is executed bears consequences that can impact the state of the virtual environment. The way the action is performed, however, does not impact the feasibility of other actions, which is why we do not take it into account when describing the scenario. An illustration of this separation into atomic actions is presented in Figure 2. In the example provided, the user grasps an object, resulting in the action `Take` being triggered. Then the user moves the object however they want to, which does not impact the state of the environment, as the object is still taken, and the hand of the user is still full. The user finishes by placing the object somewhere, resulting in the `Place` action, meaning that the place is now occupied, the hand is free and the object can be taken. Even though we are not interested in the precise path of the object, checkpoints can be defined to trigger an action. In this case, the action `Move to` is triggered when the object is placed at a given spot, but is still in the hand of the user. This kind of action can be used to force the user to place an object under a stream to wash it, or under a light to observe it thoroughly, for instance.

This definition of the actions is derived from the object-relation concept, presented in *Cause and Effects* [21], and based on two main concepts: the typed objects and the relations. The typed objects are the objects present in the environment, which are enriched with properties, or types, to make them interactible. In the object-relation paradigm, any element of the environment can be considered an object. Because of this, not only the visible inanimate objects are included: the user's avatar and more conceptual elements, such as spatial zones of the environment, can be defined as objects. The second concept is the relation, which define the actions. Instead of using the objects directly, the relations use the types placed on them. Thanks to this, a relation needs to be defined only once to be reused as many times as needed, with as many objects as possible, as long as the objects bear the necessary types.

3.2 Recording of the expert

We propose to use the *create by doing* approach to create scenarios through a recording of a user. Thanks to this approach, the authoring of a scenario becomes as natural as possible, since only a demonstration by the expert (e.g. the teacher in a nurses school) of the said procedure is needed to obtain the corresponding action sequence, in a format legible for the application. The main steps of this approach, summarized in Figure 1, are:

– **Defining the interactions in the virtual environment** Before being able to create the scenario, the types and relations must be defined, so that the environment can be really interactive. Although this part is often the role of the developer, a possible way to do it is presented in [19].

– **Run the application in a free mode** The expert is free to perform any action in the environment, with the actions being logged into a scenario. At the end of this step, the expert obtains a scenario composed of a linear sequence, listing the actions performed in order.

– **Edit the scenario** If the expert needs a more complex scenario, with branches and loops, the edition can be done using the sequences as a base. It is also possible to create several sequences to combine them as a scenario.

The main improvement provided by the *create by doing* approach is the translation of sequenced actions into a digital data. The data obtained is a scenario, in which the states correspond to the moments between the steps of the sequence recorded by the scenario generator. The transitions bear the information as to which action should be performed to make the scenario change state. In doing so, the resulting scenario is the scenario that can only be completed if a final user performed the exact same actions as the expert did during the record. Since the actions are only the important events triggered by the final user, however, the latter could be free to walk or to move an object around between each transition.

By letting the domain expert evolve freely in the environment, no constraints other than the need to be immersed in the virtual environment, is needed to create the scenario. This helps the domain expert to get a digital representation of the sequence, without having to seek help from a developer to translate the sequence. The result from this step can also be manipulated directly by the developer to add fine-grain modifications. For instance, the developer can add transitions between the scenario states, that are not triggered by an action, but rather by an event in the virtual environment. A common case is to let the scenario wait for a few seconds.

Since a scenario is composed of actions, with temporal constraints between them, it can easily be represented by an oriented graph, such as the one shown in Figure 2. Thanks to this, it is easy for the domain expert to modify the sequencing with graphical tools, with a WYSIWYG metaphor.

To let an expert create a scenario through the *create by doing* principle, we designed a set of tools, helping both in the authoring and the reuse of the scenario. In Section 4, we present a tool to create scenarios through the actions

Fig. 2. The actions recorder for a pick and place task, and the corresponding scenario

of the expert. In Section 5, we focus on the tools provided to reuse the scenario in a virtual reality application.

4 Scenario authoring tool

To enable the creation of scenarios for virtual reality applications, a domain expert needs tools that can be easily used. Based on the *create by doing* approach, we propose a tool, completely integrated in Unity, to let the domain expert create their own scenario with as few manipulations as possible.

To implement the *create by doing* approach, we needed to integrate both a model for the interactions, and a model for the scenarios. The approach in itself is not dependent from any given model, as it only defines a metaphor for the creation of the scenario. However, to implement it in our tool, we choose to base the interactions on #FIVE [4], and the scenarios on #SEVEN [9]. The main advantage of both models is that they are designed to be as versatile as possible, which allows us to create a tool usable for a large range of domains. They are also already integrated to work together in Unity, which makes the integration of the actions in the scenario easier.

In #SEVEN, the sequencing is represented by a Petri net, where the places indicate the state the scenario is in, and the transitions are triggered by changes in the environment. The main interest of this kind of representation is that the Petri net formalism is interesting for the creation of scenarios with collaboration between multiple agents. Indeed, there can be different branches of the scenario functioning in parallel, allowing several agents to share the tasks. To make the link with the virtual environment, the transitions are enriched with sensors and effectors, which respectively listen to the environment and act on it. Neither the sensor nor the effector are mandatory for the transitions: a transition without a sensor will be triggered automatically, when the state of the scenario allows it. Similarly, a transition without any effector change the state of the scenario, without any impact for the virtual environment.

#FIVE is an object-relation model based on four main concepts: the objects, the types, the relations, and the object patterns. The objects, types and relations correspond to the ones defined by the object-relation paradigm. The objects are enriched with types to make them interactible, and the relations use those types to perform the interaction. Objects pattern can be used to define more complex conditions for the objects to be used in the relations. Their main use is to define that a relation needs, for some objects, multiple types at once. When a relation is possible in #FIVE, it is instantiated in the form of a *realization*. The realization is directly linked to the objects manipulated, as opposed to the relation.

During the recording of the expert, all the relations in the environment are allowed. Thanks to this, the expert can manipulate the objects as much as they want. Each time the expert performs an action, the corresponding relation is recorded in the scenario under the form of a new transition, creating a scenario as illustrated in Figure 3. This phase generates a sequence, with each transi-

tion corresponding to a relation. The created transition contains two pieces of information: the relation executed by the expert, and the objects involved in it.

The recording is done thanks to a specific scenario, which simply waits for a relation to be executed. Once it is done, the effector on the transition writes a new part for the scenario under construction. Then, the scenario just loops to the original place to be ready to listen to a new relation.

It is to note that the actions done by the user during the recording impact the state of the environment. Because of this, the environment is reset when the expert ends the recording. Before that, the recorded scenario cannot be re-injected in the application, to preserve the coherency of the virtual environment.

To help the expert visualize the scenario, a graphical editor is provided in Unity. This editor shows the scenario as illustrated in Figure 3, with tools to modify it. During the creation of the scenario, the editor can display the scenario in real-time, as the transitions and places are added according to the actions done by the expert. The graphical editor can be used to modify the sequencing of the actions, by deleting and adding links between the places and the transitions. Of course, the expert is also free to modify the sensors and effectors on the transitions, although it is more pertinent for a developer to do those finer modifications. With the graphical editor, the domain expert can also easily combine multiple scenarios into a single, more complex one, by adding the sequences as branches in the final scenario.

5 Scenario reuse

Once the scenario has been created, the goal of the domain expert is of course to use it in an application. Similarly to the creation of the scenario, its reuse in the environment is difficult without the necessary knowledge in coding. To allow the expert to reuse the scenario obtained through the *create by doing* approach, we propose some tools to ease the workflow of re-integrating the scenario in the virtual environment. In Section 5.1, we first present how the expert can reuse the scenario for a simulation with a human user. In Section 5.2, we describe how a virtual agent can be connected to the scenario in an application. We can list four different reuses available for the scenario,: the *learn by doing*, the *monitor by seeing*, the *collaborate by doing* and the *learn by seeing*.

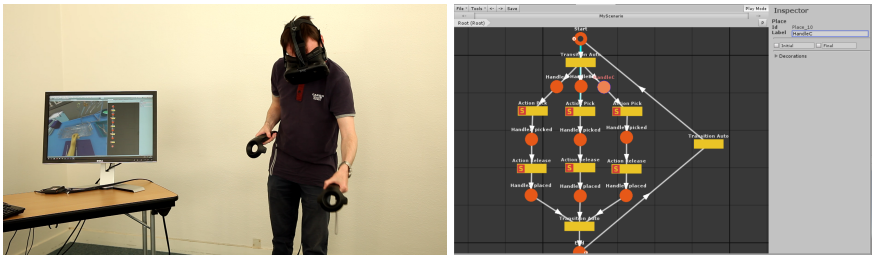

Fig. 3. On the left, an expert creates a scenario. On the right, the scenario editing tool

For more customization when reusing the scenario, the developer can also directly access the resulting scenario, which is in the #SEVEN format. With it, the developer can either use the graphical editor, along with the Unity scene, or even modify the file directly. An example of customization is the creation of variations for the scenario to use it under different pedagogy conditions.

5.1 Reuse with a human user

The main objective of the scenario is often to be reused by a human user, to have a set of tasks to complete in the virtual environment. The scenario created by the *create by doing* approach is, by nature, usable by the scenario engine, since it is the scenario engine itself that generates it. Thanks to this, the scenario can be easily re-injected in the application. However, to make a good use of this scenario, we propose two complementary uses, involving a human user: the *learn by doing*, and the *monitor by seeing*.

Learn by doing The most obvious reuse for a scenario is to re-inject it in the virtual environment, to guide the actions of a user. Indeed, scenarios in virtual reality are common for applications such as serious games [3]. In those applications, the interactivity is important, as the user feels more immersed, and hence more involved in the learning [11]. This guidance offered by the scenario is particularly useful for the training to procedures, we call this reuse the "learn by doing".

To make the *learn by doing* possible, the domain expert needs only to define the scenario they generated as the scenario used for the simulation. For this, the graphical editor we provide proposes a feature to automatically set the displayed scenario as the simulation's scenario. Thanks to this, the scenario will automatically be used when the application is launched again.

During the training to a procedure, different levels of guiding can be provided. In order to simulate this guidance, we provide highlights in the environment, with predefined settings depending on the kind of guidance wanted by the expert (more can be defined by tuning the behaviours of the guiding elements):

– **No guidance** In some cases, for instance when the application is used to test the knowledge of a student, no guidance should be offered by the environment. In this case, all the highlights are disabled. However, the scenario is still present in the application, and can be used to notify the student about the success or failure of the procedure. Indeed, when the scenario arrives in a final state, this state can be used to define whether or not the student managed to perform the procedure successfully.

– **Step-by-step guidance** For a first explanation of the procedure, it can be useful to show to the user what they should do to continue the scenario. For this, a setting is proposed to highlight the actions that would trigger the available transitions in the scenario, (i.e. the transitions for which the upstream places all have a token to use). For each pertinent action to highlight, the objects to manipulate are highlighted as shown in Figure 4. Since the highlight alone is

not enough to explain which action should be done with the object, the expert may associate each relation of the environment with a color, and with a different kind of highlight (contouring, flare, ...). This way, the objects will be highlighted differently depending on the actions to perform. To avoid giving too much information, only one highlight is displayed per object, depending on its priority (set for each action). Depending on the expert's choice, the actions doable by the user can either be limited to the ones proposed by the scenario or not.

–**Limited guidance** For specific conditions, such as a rehearsal of the procedure, a lighter guidance is needed. To provide this kind of guidance, we propose to highlight all the interactible objects in the environment, regardless of the specific interactions indicated by the scenario. This indicates what objects can be manipulated, but nothing more.

Monitor by seeing When executing the scenario, a trainer can have difficulties to see what the user is doing. It is especially difficult when the user goes fast or when there are many objects in the environment.

To help the trainer determine what the user is doing, and to check if the user is correctly following the procedure, we propose a means to monitor the execution of the scenario. To do this, we propose to display the scenario, in the graphical editor integrated in Unity, during the runtime of an application.

When the application is playing, the scenario is displayed differently in the editor, to show its current state. More precisely, the tokens of the Petri net are moved according to the transitions that are triggered by the user. When a transition is available in the scenario, it is also highlighted in the editor. That way, it is easy for the observer to see how the scenario is being executed.

This feedback of the running scenario can be combined with the visual feedback showing what the user sees. That way, a more complete information can be given to the observer.

5.2 Reuse with a virtual actor

To help with the use of the scenario, the application may need one or several virtual agents to perform some tasks. To help the domain expert use a virtual

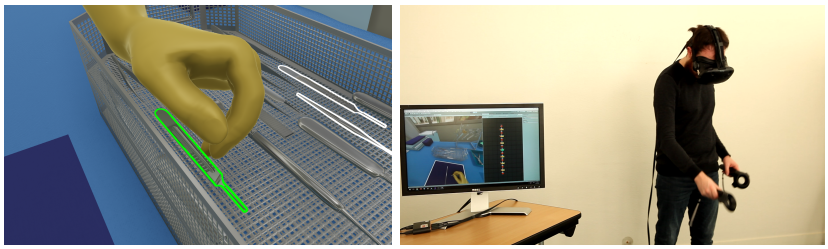

Fig. 4. On the left, a scalpel being highlighted to show that the user can take it. On the right, a trainee learning the scenario by interacting with the environment

agent connected to the scenario in their virtual reality application, we propose a complete package to connect a humanoid to the scenario.

We designed this package so that it automatizes most of the behaviour of a virtual agent when it is linked to a scenario. Thanks to it, a virtual actor is able to move to a given location, grasp an object using inverse kinematics (with FinalIK [3]), and execute actions from the scenario.

With our package, the virtual agent executes the scenario as follows:

- The virtual agent lists the available actions given by the current state of the scenario. By default, the virtual agent can execute any available action. However, it is possible to limit its capacities, by adding roles to the actors (through their GameObject), and putting the same roles on the transitions: a transition with roles can only be triggered by an actor with the same roles. All of it can be done directly in Unity, through the scenario editor.
- For each one of those available actions, the virtual agent checks if the action can be done considering the state of the objects involved in the action. For instance, if the action proposed by the scenario is to take an object, and the agent already has its hands full, the action should not be taken into account. Thanks to the first two steps, a list of the actions that can be considered both useful and feasible is obtained.
- If there are more than one actions available for the virtual agent, one of them is chosen and executed. The choice of the action amongst the available ones is done according to a heuristic set by the expert. By default, the choice is done randomly, but the use of more detailed heuristics (e.g. computing the shortest path regarding the number of actions or choosing the closest items) allows to modify the behaviour of the virtual agent. For instance, it can try to minimize the number of actions needed to complete the task.

To make the virtual agent look more credible, all of its actions are animated. The tools provided can work with any humanoid, as long as it is compatible with Unity's humanoid system. To do this, we add some information to the objects and relations, to tune the avatar's behaviour accordingly. There are two kinds of features that can be used: additional methods in the relations, and descriptors for the objects.

Each relation defines the animation for its execution. When executing the relation, the virtual agent reads the necessary information in the relation, and performs the corresponding animation. To modify the animation for a given relation, a developer can decide how the avatar is supposed to behave, and modify the code of the relation accordingly.

To define more precisely how the object should be grasped by the avatar, we add descriptors on the objects. Those descriptors, which can be used for the virtual agent as well as for the user's avatar, define the pose of the hand when the object is grasped. It describes which hand can be used to take the object (either the left one, the right one, any hand or both hands simultaneously). For each hand, the pose is defined by describing the positions of the joints for each

[3] https://assetstore.unity.com/packages/tools/animation/final-ik-14290

finger. For an easier setting of this descriptor, the rig can be displayed in Unity, to see how the hand is supposed to go around the object, like in Figure 5.

The object descriptors also define how the hand transitions between poses when it does the animation. Indeed, we can distinguish two states for the hand: with a pose or loose. Depending on the object and the action, the hand either take a pose only when it is close to the object (for instance, to push a button), take a pose and keep it (to grasp an object), release the pose (to release an object) or stay as it is during the whole animation.

Thanks to the descriptors, and to the scenario system for the virtual agent, it is easy for the domain expert to get a virtual agent to execute some tasks in a scenario. We can distinguish two main uses for a virtual agent, controlled by a scenario, in virtual reality: the *collaborate by doing*, and the *learn by seeing*.

Collaborate by doing The first, called the *collaborate by doing*, is the most common one. It consists in using one or several virtual agents to collaborate with the user, in order to help/disturb them during the procedure.

Since the same scenario is used by both the human user and the virtual agent, the collaboration is implicitly defined in the scenario itself, with as many agents as wanted. If needed, the domain expert can tune this collaboration, by appointing roles to each actor. Those roles must also be placed on the transitions in the editor: only an actor with the needed roles can trigger the transition. This does not prevent the actors from executing the actions that would trigger the transition. However, the virtual agents only list the actions for which they have the necessary roles. Because of this, they are "blind" to the rest of the scenario, and will not execute actions that do not correspond to their part.

Learn by seeing The virtual agent can also be useful on its own, without any human user in the virtual environment. Since it may not be always easy for a trainer to get the material needed for a demonstration, we propose to use the replay of the scenario by a virtual agent as a demonstration. The scenario defines a complete set of tasks, in order, which can be replayed by a virtual agent in the scenario, the same way as if the agent was collaborating with a user.

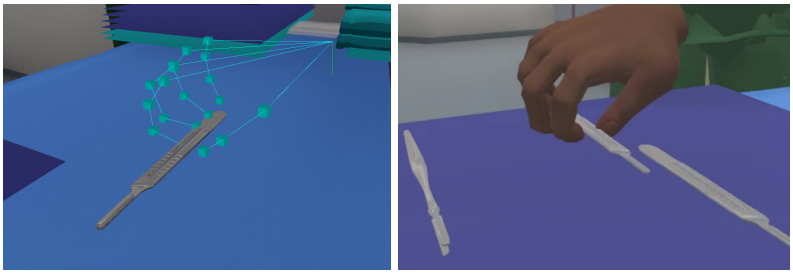

Fig. 5. The descriptors for a hand pose, and the resulting hand grasping the scalpel

Once the scenario is set in the virtual reality application, the domain expert only needs to launch the application, without any human user, to let the virtual agent perform the actions. A trainee can then observe the virtual agent perform the scenario to learn the procedure. Since the virtual agent is performing in real-time, in Unity, the trainee can easily change the viewpoint or be immersed in the scene to get the opportunity to observe from the most pertinent viewpoint at any moment. In the latter case, the trainee will just see the virtual agent perform the actions, and will be free to move around, with the possible help of a teacher to indicate what is interesting to look at. The demonstration performed by the virtual agent can also be easily recorded into a video, by recording the feedback given by the cameras in the Unity scene. This can be done with pre-integrated plug-ins in Unity, or by capturing the feedback given on screen. The recording can include a camera placed on the virtual agent's head, to give a first person view of the demonstration.

6 Use cases

To demonstrate our approach and tools, we designed two use cases: a medical use case in Section 6.1, and a archaeology one in Section 6.2. Since the tool is designed to be independent of the application, it has been used as is for both use cases.

6.1 Surgical procedure training

In the medical domain, the knowledge of procedures is a crucial part of the success for a surgical intervention. Indeed, in order not to cause problems or slow down the whole group, each member of the team must know their tasks. When learning procedures, virtual reality can be a powerful tool to learn how to work in a team [24]. To help the students learn the procedures for the preparation of a room before the surgical intervention, we designed, in collaboration with a medical team, an application for scrub nurses. It was also experimented in a nurses school, to evaluate its potential for training. In this application, the scrub nurse in training must prepare an operating table before the intervention.

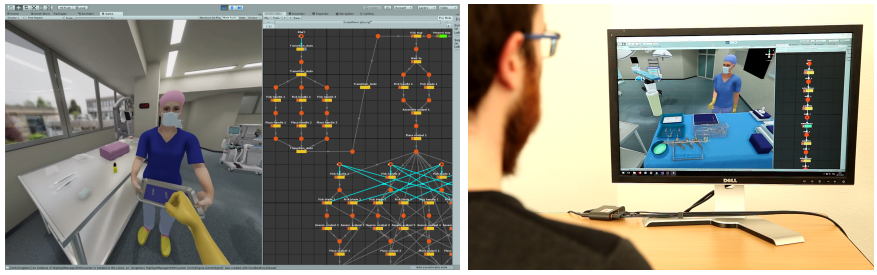

Fig. 6. A user is collaborating with a virtual nurse

In this use case, the trainee has to take some instruments, and place them on specific spots on the table, so that they can be rapidly found during the intervention. Some of the instruments must also be prepared during this preparation phase, such as scalpels that must be assembled. The preparation of the table differs for each intervention, which is why it is important to be able to create new scenarios easily. In one of the scenarios we tested, a part of the procedure makes the trainee assemble three scalpels (composed of a handle and a blade), by first putting the handle on the table, taking the blades, and placing back the assembled scalpels. For this task, the trainee must collaborate with a virtual agent, who brings the blades, which must be assembled with the handles, and placed on the table. The setup for this application is illustrated in Figure 7.

In this application, we already integrated all the possible interactions in the virtual environment beforehand, with #FIVE. Thanks to this, only the creation of the scenario for each procedure remains to be done by the domain expert. To do this, the domain expert can simply perform the whole preparation, which results in the complete scenario.

In this scenario, some of the actions can be done in any order. For instance, there is no particular constraint for the assembling of the scalpels: the nurse can assemble them in any order. Because of this, the graphical editor was used to modify the sequencing of the actions. Instead of having a linear sequence, some of the actions were put in parallel branches, so that each scalpel assembly corresponds to a separate branch.

The help of a virtual agent was also needed. Because of this, we defined that the virtual agent could only bring the plate containing the scalpel blades to the user, once the handles are put on the table. This was defined using the roles of the scenario: the virtual agent cannot do the first part of the scenario, and is hence waiting for the user to do it. This way, the virtual agent has a specific role in the scenario, which does not collide with the trainee's actions.

Finally, we used the different levels of guidance in the scenario to help trainees to learn the table preparation procedure. The first version presented to them used a strong guidance, to highlight the following actions that were useful for

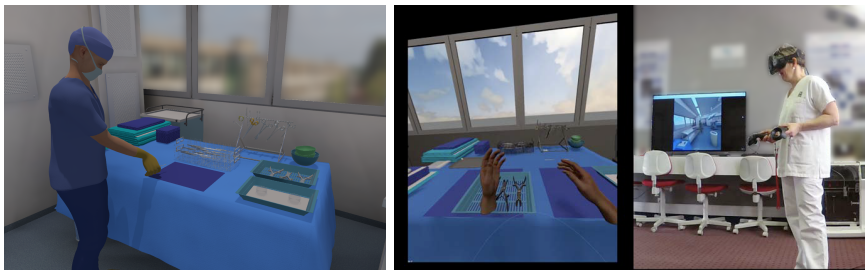

Fig. 7. On the left, the preparation table in the virtual operating room. In the middle, the first person view of the environment. On the right, a real expert scrub nurse interacting with the application.

the preparation of the table. Then, another version was presented to them, with only a limited guidance: all the possible actions were highlighted. During the execution of the procedure, a trainer can see the vision of the trainee, as well as the scenario being updated according to the trainee's actions.

While the first application allowed us to reuse the scenario for the *collaborate by doing* and the *monitor by seeing* cases, we were also able to use the scenario in the *learn by seeing* case. This second use of the scenario was done to demonstrate the assembly of the scalpels through a replay of the procedure by a virtual actor. To do this, we replaced the human user by a virtual agent, and recorded it assembling of the scalpels. The resulting replay shows the complete assembly, and can easily be presented to a group of students to explain the procedure, without having to be in the operating room, or to recreate it in a classroom.

6.2 Archaeological virtual tour

To test the genericity of our approach, we decided to use it with a second domain, namely archaeology. Once an archaeological excavation has been done, it is often difficult for the archaeologists to diffuse the newfound knowledge to the general public. To help in doing so, the archaeologists may organise guided tours on the excavation site at the end of the excavation, to explain what has been found, with the visual supported provided by the site itself. To be able to visit it after the site is closed, we designed a virtual tour application. This use case provides another *learn by doing* situation, with different stakes.

In this particular use case, we propose to visit the site of Beg-Er-Vil, in the west of France. The scenario for the virtual tour was designed with the collaboration of an archaeologist who worked on the excavation. This virtual tour allows a tourist to be immersed in the site as it was during the excavation, and to explain the discoveries, as well as the work of the archaeologists. To ensure a better understanding of the information presented in the virtual site, a precise path is defined, which brings the visitor to pertinent points of interest, with explanations at each point.

In order to keep the application simple for the visitors of the virtual site, we decided to limit the possible interactions. First, the user can move around the site through teleportation. On the external part of the site, specific teleportation targets are already placed, to provide the means to easily access each part of the environment. In the central part of the site, where the density of POIs is higher, the teleportation can be done to any location of the zone. They can also display and hide textual information for each point of interest in the virtual environment, in the form of a panel; and modify some parameters of the simulation.

When the visitors starts the application, they view something akin to what is presented in Figure 8. For the creation of the scenario, the archaeologist is immersed in the virtual site. For the navigation in the site, teleportation targets are automatically placed so as to cover the maximum space in the environment. In the core of the site, however, the user is free to teleport anywhere in the zone, as the interesting spots are more condensed in that zone.

In the virtual environment, the archaeologist can place points of interest in the environment during the creation of the scenario. The creation of the POIs is done by adding a new point in the environment, and telling the content of the panel, which is automatically converted to text. Since the creation of the POIs should not be integrated in the final scenario, the relations associated to that interaction are specifically declared as ignored by the scenario recorded, in order to keep only the pertinent interactions.

At any moment, the archaeologist can also modify some parameters of the simulation. For instance, since this site is on the coast, the sea level can be modified, to show how it was at different times during the occupation of the site. Those modifications of parameters can be really helpful to make the visitors aware of the changes that occurred since the site was left by its occupants, and to understand their living conditions at the time.

Once the archaeologist recorded the scenario, the reuse is simply done by the re-injection of the scenario in the application. Thanks to it, a visitor can then use the application to virtually visit the archaeological site. For this reuse, we decided to use a *learn by doing* approach. To help the visitors follow the path designed by the archaeologist, a step-by-step guidance is used, with highlights showing where the user should go next, and with which POIs they should interact. However, their actions are not limited to the ones provided to the scenario, to let a user go back and visit freely other locations is they want to.

7 User feedback

In order to evaluate our approach, we asked 5 experts (1 from the medical domain and 4 archaeologists). The participants were presented to the environment with a first, simple example (changing the wheel of a car). Once they were accustomed to it, they moved to the use case corresponding to their domain of expertise: the medical staff was presented with the operating room, and the archaeologists were immersed in an office. The goal was to present an environment in which

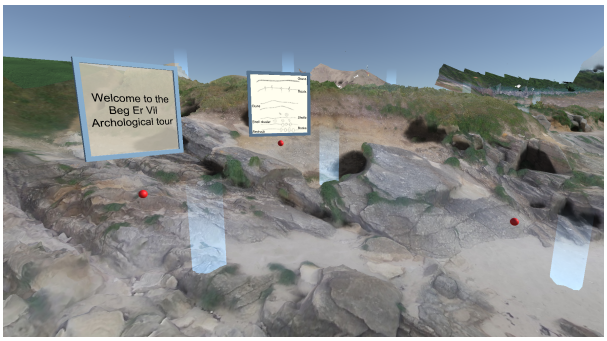

Fig. 8. User's view at the start of the virtual tour (with the teleportation points)

the user could be at ease, with exactly the same interactions between the two examples.

In both cases, the desk was filled with different tools, and the participant was asked to create the scenario corresponding to the following procedure; place some of those tools at a given place, in a certain order. At the end of the session, we showed them the resulting scenario in the editor, and let them modify it as they wanted, until they felt accustomed to the editor. Finally, we asked them to fill the UTAUT2 questionnaire [27], along with the NASA-TLX [14], the SUS questionnaire [26], a Simulator Sickness questionnaire [15], and a Personal Inovativeness questionnaire [1] (the final questionnaire is provided with this paper).

Overall, the experts feedback was strongly positive. They considered that the tool was useful ("*I think that this tool is useful to create scenarios*" $\rightarrow$ m=7, σ=0.447, 7 being the best score), simple ("*Few efforts are needed to be at ease with the tool*" $\rightarrow$ m=6, σ=0.5477) and fun to use (m=7, σ=0.447). All of them managed to obtain the expected scenario, without much effort. However, there were also some negative comments, mostly directed at the possibility to use virtual reality in their work (cost, accessibility). The participants who felt this way were also the same that scored the worst score for the personal innovatiness questionnaire. Also, one of them noted that he thought that the tool could be difficult to use for more complex scenarios in the open comments section. Still, all the participants found the tool interesting and really easy to use, and were satisfied to have been able to create a scenario without direct help.

8 Conclusion and future works

In this paper, we presented a new workflow for the creation of scenario-driven applications in virtual reality. Thanks to a *create by doing* approach, an expert with no particular coding skills is able to generate a complete scenario, and to reuse it in an application. This reuse can be done with a single user, or with virtual agents controlled by the scenario, to obtain different ways of presenting the scenario to a user. Thanks to this, a same scenario can be easily created and reused for either a *learn by doing*, a *learn by seeing*, a *collaborate by doing*, or a *monitor by seeing* application.

Fig. 9. The environment for the experiment

For the complete creation and reuse of the scenario, we provide three important tools for the domain expert. The first tool, also briefly presented in [19], allows the domain expert to create a scenario based on a demonstration of it done directly in the virtual environment. The second one is a tool to use the scenario to guide a user in the virtual environment, with different levels of guidance according to the kind of application wanted. Finally, the proposed methodology is supported by a tool to connect a virtual agent to a scenario. This tool allows the expert to define how a virtual agent should behave, to either perform a part of the scenario or demonstrate it completely.

We will enhance this approach in several ways in future works. First, the definition of scenarios with variations (e.g. different orders) could be simplified. For now, only the objects used during the recording are integrated, without any possibility of variation. To add this variation, we propose to record the expert multiple times, and to synthesize the result given by the set of observations to get the final scenario. Mathematical models such as test and flip synthesis [6] could also be used to detect patterns in the scenarios, allowing automatic creation of variations.

A second improvement could be the integration of pedagogy into the reuse of the scenario. While, in this paper, we focused on bringing tools for the experts, leaving the pedagogical aspect to the teachers and pedagogy experts, it could be interesting to bring more efficient tools to tune the experience of the learners, by taking into account known counterexamples for instance.

A final improvement could be to give tools to edit efficiently other kinds of sensors and effectors in the scenarios, so as to give the expert more ways to handle the creation of the application. Examples of such sensors and effectors are timers, or effectors that could change roles dynamically: while those are integrated in the editor, it is still difficult for the expert to understand them.

References

1. Agarwal, R., Karahanna, E.: Time flies when you're having fun: Cognitive absorption and beliefs about information technology usage. MIS quarterly pp. 665–694 (2000), https://www.jstor.org/stable/3250951?seq=1
2. Angros, Jr., R., Johnson, W.L., Rickel, J., Scholer, A.: Learning domain knowledge for teaching procedural skills. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3. pp. 1372–1378. AAMAS '02, ACM, New York, NY, USA (2002), http://doi.acm.org/10.1145/545056.545134
3. Bailenson, J.N., Yee, N., Blascovich, J., Beall, A.C., Lundblad, N., Jin, M.: The use of immersive virtual reality in the learning sciences: Digital transformations of teachers, students, and social context. Journal of the Learning Sciences **17**(1), 102–141 (2008), https://doi.org/10.1080/10508400701793141
4. Bouville, R., Gouranton, V., Boggini, T., Nouviale, F., Arnaldi, B.: #FIVE : High-Level Components for Developing Collaborative and Interactive Virtual Environments. In: Proceedings of Eighth Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS 2015), conjunction with IEEE Virtual Reality (VR). Arles, France (3 2015), https://hal.inria.fr/hal-01147734

5. Brom, C., Šisler, V., Holan, T.: Story Manager in 'Europe 2045' Uses Petri Nets, pp. 38–50. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-77039-8_4
6. Caillaud, B.: Surgical Process Mining with Test and Flip Net Synthesis. In: Bergenthum, R., Carmona, J. (eds.) Application of Region Theory (ART). pp. 43–54. Barcelona, Spain (Jul 2013), https://hal.inria.fr/hal-00872284
7. Chan, J.C.P., Leung, H., Tang, J.K.T., Komura, T.: A virtual reality dance training system using motion capture technology. IEEE Transactions on Learning Technologies **4**(2), 187–195 (April 2011), https://doi.org/10.1109/TLT.2010.27
8. Chevaillier, P., Trinh, T.H., Barange, M., De Loor, P., Devillers, F., Soler, J., Querrec, R.: Semantic modeling of Virtual Environments using MASCARET. In: 2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS). pp. 1–8 (3 2012), dx.doi.org/10.1109/SEARIS.2012.6231174
9. Claude, G., Gouranton, V., Bouville Berthelot, R., Arnaldi, B.: Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment. In: Nojima, T., Reiners, D., Staadt, O. (eds.) ICAT-EGVE, International Conference on Artificial Reality and Telexistence, Eurographics Symposium on Virtual Environments. pp. 1–4. Bremen, Germany (Dec 2014), https://hal.archives-ouvertes.fr/hal-01086237
10. Cremer, J., Kearney, J., Papelis, Y.: HCSM: a framework for behavior and scenario control in virtual environments. ACM Transactions on Modeling and Computer Simulation (TOMACS) **5**(3), 242–267 (1995), http://dx.doi.org/10.1145/217853.217857
11. Fletcher, J.D.: Does this stuff work? a review of technology used to teach. TechKnowlogia **Jan-Mar**, 10–14 (2003)
12. Gerbaud, S., Mollet, N., Arnaldi, B.: Virtual Environments for Training: From Individual Learning to Collaboration with Humanoids, pp. 116–127. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-73011-8_14
13. Green, T.R.G., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages & Computing **7**(2), 131–174 (1996), http://www.sciencedirect.com/science/article/pii/S1045926X96900099
14. Hart, S.G., Staveland, L.E.: Development of nasa-tlx (task load index): Results of empirical and theoretical research. In: Advances in psychology, vol. 52, pp. 139–183. Elsevier (1988), https://www.sciencedirect.com/science/article/pii/S0166411508623869
15. Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology **3**(3), 203–220 (1993). https://doi.org/10.1207/s15327108ijap0303_3, https://doi.org/10.1207/s15327108ijap0303_3
16. Klopfer, E., Perry, J., Squire, K., Jan, M.F., Steinkuehler, C.: Mystery at the museum: a collaborative game for museum education. In: Proceedings of the 2005 Conference on Computer support for collaborative learning. pp. 316–320. International Society of the Learning Sciences (2005), http://dl.acm.org/citation.cfm?id=1149293.1149334
17. Klopfer, E., Squire, K.: Environmental detectives—the development of an augmented reality platform for environmental simulations. Educational Technology Research and Development **56**, 203–228 (04 2007)

18. Lamarche, F., Donikian, S.: Automatic orchestration of behaviours through the management of resources and priority levels. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3. pp. 1309–1316. ACM (2002), https://doi.org/10.1145/545056.545124
19. Lécuyer, F., Gouranton, V., Reuzeau, A., Gaugne, R., Arnaldi, B.: Create by doing – Action sequencing in VR. In: Gavrilova, M., Chang, J., Thalmann, N.M., Hitzer, E., Ishikawa, H. (eds.) Advances in Computer Graphics. pp. 329–335. Springer International Publishing", Cham (Jun 2019). https://doi.org/10.1007/978-3-030-22514-8_27
20. Lee, G.A., Nelles, C., Billinghurst, M., Kim, G.J.: Immersive authoring of tangible augmented reality applications. In: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality. pp. 172–181. ISMAR '04, IEEE Computer Society, Washington, DC, USA (2004), http://dx.doi.org/10.1109/ISMAR.2004.34
21. Lugrin, J.L., Cavazza, M.: Making sense of virtual environments: action representation, grounding and common sense. In: Proceedings of the 12th international conference on Intelligent user interfaces. pp. 225–234. ACM (2007), https://doi.org/10.1145/1216295.1216336
22. Marfisi-Schottman, I., George, S., Tarpin-Bernard, F.: Tools and Methods for Efficiently Designing Serious Games. In: European Conference on Games Based Learning, ECGBL. pp. 226–234. Copenhagen, Denmark (2010), https://hal.archives-ouvertes.fr/hal-00953318
23. Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE Intelligent Systems **17**(4), 39–47 (Jul 2002), http://dx.doi.org/10.1109/MIS.2002.1024751
24. Mitchell, L., Flin, R., Yule, S., Mitchell, J., Coutts, K., Youngson, G.: Evaluation of the scrub practitioners' list of intraoperative non-technical skills (splints) system. International Journal of Nursing Studies **49**(2), 201 – 211 (2012), http://www.sciencedirect.com/science/article/pii/S002074891100335X
25. Paiva, A., Machado, I., Prada, R.: Heroes, villians, magicians, & dramatis personae in a virtual story creation environment. In: Proceedings of the 6th International Conference on Intelligent User Interfaces. pp. 129–136. IUI '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/359784.360314
26. Slater, M., Usoh, M., Steed, A.: Depth of presence in virtual environments. Presence: Teleoperators & Virtual Environments **3**(2), 130–144 (1994), https://doi.org/10.1162/pres.1994.3.2.130
27. Venkatesh, V., Thong, J.Y., Xu, X.: Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. MIS quarterly **36**(1), 157–178 (2012), http://www.academia.edu/download/36422124/Venkatesh_utaut2.pdf